Centre de recherche sur les transports (C.R.T.)
Centre for Research on Transportation
CRT-2004-28

SCHEDULING A HOT ROLLING MILL

by

Daniel De Ladurantaye
Michel Gendreau
Jean-Yves Potvin

December 2004

Université de Montréal
C.P. 6128, succursale Centre-ville, Montréal QC H3C 3J7 Canada
Téléphone: (514) 343-7575 / Télécopieur: (514) 343-7121

crt@crt.umontreal.ca
www.crt.umontreal.ca

HEC ÉCOLE POLYTECHNIQUE MONTRÉAL Université de Montréal

PUBLICATION
CRT-2004-28

SCHEDULING A HOT ROLLING MILL

by

Daniel De Ladurantaye[1]
Michel Gendreau[1]
Jean-Yves Potvin[1]

The research was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC), and by Keops Technologies Inc.

1. Département d'informatique et de recherche opérationnelle and Centre de recherche sur les transports, Université de Montréal, C.P. 6128, succursale Centre-ville, Montréal, Canada H3C 3J7 (daniel.de.ladurantaye@umontreal.ca; michelg@crt.umontreal.ca, jean-yves.potvin@umontreal.ca)

Dépôt légal – Bibliothèque nationale du Québec
Bibliothèque nationale du Canada, 2004

105-01987/

Résumé. Cet article présente une heuristique pour résoudre un problème d'ordonnancement pour un laminoir à chaud dans l'industrie de l'aluminium. Une des difficultés de ce problème est la forte dépendance entre les fours d'homogénéisation et le laminoir, ce qui doit être tenu en compte lors de la conception de l'heuristique. Cette dernière considère également plusieurs contraintes technologiques comme les transitions de dureté entre les différents alliages, l'usure des rouleaux, la compatibilité entre les codes de chauffage et les transitions de largeur. L'objectif est de minimiser les temps morts sur le laminoir et les pénalités pour les violations des contraintes molles (relatives à la qualité des produits finis). L'heuristique est divisée en deux phases. Premièrement, des lots de lingots devant être chauffés ensemble dans les fours sont générés. Ces lots, appelés blocs, sont ensuite séquencés sur le laminoir. Des résultats numériques sur des données réelles sont présentés.

Mots clés. Ordonnancement, laminoir à chaud, fours d'homogénéisation, lots, heuristiques.

Abstract. This paper presents a heuristic algorithm to schedule a hot rolling mill in the aluminum industry. One problematic issue is the tight coupling between the homogenizing furnaces and the mill, which needs to be integrated into the heuristic design. The latter also takes into account standard technological constraints like alloy hardness transitions, roll wear, homogenization code compatibilities and width transitions. The objective is to minimize the idle time on the mill and penalties for soft constraint violations (which relate to production quality). The heuristic is divided into two phases. First, batches of ingots are constructed for the furnaces. These batches, called blocks, are then sequenced on the mill. Numerical results are reported on test instances derived from real-world data.

Keywords. Scheduling, hot rolling mill, homogenization furnaces, batches, heuristics.

1 Introduction

In a standard rolling plant there are three main shops, the foundry, the hot line and the cold line. In the foundry, recycled pieces of aluminum are melted in large furnaces. During this operation, different elements like magnesium or zinc are added to the mixture to obtain a specific alloy. Afterwards, the metal in liquid form is poured into a casting pit to produce ingots. Different dimensions are obtained by changing the casting table. When ingots are solidified and cooled, they are transferred to the hot line. First, they are scalped to eliminate impurities that tend to accumulate on the edges. Then, ingots are heated in homogenizing furnaces to obtain desired mechanical properties and to prepare them for the next operation, where they are rolled in a hot mill at near 600^oC. This operation reduces their thickness to a fraction of an inch and produces long sheet coils. In the cold line, coils are processed by different machine centers, like the cold mill, annealing furnaces, and tension leveller to obtain their final desired properties. The sequence of operations often depends on the product and can change in course of operations, as different sequences are equivalent.

In this paper, we focus on the hot mill, in particular the homogenizing furnaces and the rolling mill. A good synchronization between the furnaces and the mill is very important because any waste time on the hot mill is costly in terms of productivity and energy savings (as the mill must be maintained at the right temperature). On the other hand, the production must satisfy quality standards. Therefore, a good trade-off between quality and efficiency must be found. To complicate the matter, the process imposes specific constraints on the solution. Some must absolutely be satisfied (*hard constraints*), while others can be relaxed by adding penalties to the objective (*soft constraints, preferences*). The goal is to find a good compromise among these conflicting objectives, while satisfying hard constraints.

In the following, Section 2 first describes the problem under study. Then, a short literature review is presented in Section 3, followed by a description of our heuristic algorithm in Section 4. Numerical results are reported in Section 5. Finally, Section 6 concludes.

2 Problem description

In a standard rolling plant, ingots are casted in the foundry, rolled into coils on the hot line and then rolled at room temperature on the cold line. The hot line contains two main machine centers: one with the homogenizing furnaces and the other with the mill, in that order. They are described in the following.

2.1 Homogenizing furnaces

The goal of homogenization is to thermically treat ingots to obtain desired mechanical properties and to have them at the right temperature for the hot rolling operation that follows. The heating time varies a lot and can take between 10 and 50 hours, depending on the homogenization code and furnace type. Each ingot has its own code and all codes in a batch must be compatible. The code with the longest duration determines the batch code and the heating time. Although this is not ideal, ingots can also be kept longer in the furnace as long as a predefined maximum time associated with each code is not exceeded.

The plant considered in this study has one pusher and two soaking pits. These two types of furnaces are described below.

Soaking Pit

The capacity of a soaking pit varies between 15 and 20 ingots depending on their dimensions. Ingots are loaded vertically in the pit and a minimal spacing between them is required to ensure uniform heating. As the capacity depends on ingot dimensions, a bin packing problem should be solved to fill a furnace with minimum residual capacity. In practice, pre-engineered tables that indicate capacity versus ingot dimensions are used. Although some manual adjustments are needed when a batch is made of ingots of different sizes, this approximation is in general satisfactory.

When loading is completed, the furnace is started and all ingots are heated at the same time. After the minimal heating time is reached, it is possible to leave ingots in the furnace, as long as the maximum heating

time is not exceeded. Beyond that limit, ingots are considered as scrap. It is thus primordial to achieve a good synchronization between the furnaces and the hot mill. It should also be noted that all ingots in a batch must be rolled before the furnace can be used again.

Pusher

As opposed to soaking pits, pushers are divided into independent compartments and can be fed with ingots on a more continuous basis. Basically, ingots are loaded in the first compartment and are pushed from one compartment to the next at time intervals that depend on the homogenization code. When the current compartment is empty, it is possible to load it immediately, if the code is compatible with the previous one, or to leave it empty and fill the next one, otherwise. It is thus beneficial to minimize the number of transitions between non-compatible codes.

Pushers can also work in batch mode, like soaking pits. That is, all compartments are filled with compatible ingots and are heated at the same time. This is how the pusher is modeled in this study. The capacity of the pusher is 25 ingots (5 compartments of 5 ingots each), independently of the ingot dimensions.

Hard and soft constraints

For each furnace type, there are different preferences and constraints, which are used to assess production quality. This information is kept in tables that are used daily. Here is the list of hard constraints that must be taken into account during homogenization:

1. *Min/max dimensions*: allowable dimension ranges for an ingot.
2. *Authorized codes*: incompatibilities between homogenization codes and furnace types.
3. *Code compatibility*: incompatibilities between codes in the same batch.
4. *Min/max heating duration*: heating duration interval that depends on the homogenization code of the batch.

For preferences, a penalty must be paid if they are not satisfied. They are:

1. *Codes and products*: Preferences may be expressed for a particular furnace type, depending on the homogenizing code and product type.

2.2 Hot rolling mill

Hot rolling considerably reduces the thickness of a metal piece by pushing it many times through two extremely hard cylinders or rolls that spin at high speed. As its name points out, this operation must be done at high temperature to ease thickness reduction. Ingots are unloaded one by one from the furnace and are directly rolled to avoid excessive cooling. Although a plant might have many rolling mills, a single mill is considered in this study.

Alloy groups and roll life

Alloys are classified into hardness groups from softer to harder. Here, we only have two groups (soft and hard) but the same ideas can be applied for more groups. Each alloy group has its own roll wear coefficient. This constant coefficient is the average amount of wear that a particular ingot of this alloy group causes when it is rolled. Rolls must be changed when they are too worn; the time between two consecutive roll changes is called a roll life.

Hard and soft constraints

To meet quality standards during rolling, some rules must be followed. Here is the list of hard constraints that must be taken into account by the hot rolling mill:

1. *Wear interval*: Each ingot of a particular hardness group has a wear interval within which it can be rolled. These intervals are not disjoint in general.

The list of preferences for the hot rolling mill is the following:

1. *Hardness transition*: as much as possible, soft ingots must be rolled before hard one in the same roll life because soft alloys require less damaged rolls.
2. *Width*: ingots should be processed in non decreasing order of width to avoid damages to the sheet edges (due to roll deformation).
3. *Final thickness or gauge*: ingots should be rolled in non decreasing order of gauge. Furthermore, large transitions between consecutive ingots should be avoided. This is because abrupt transitions require manual adjustments on the mill.

In practice, it is almost impossible to satisfy all preferences because they are often contradictory. Furthermore, the sequence on the rolling mill highly depends on the homogenization furnaces, since ingots must be heated before they are rolled.

2.3 Orders

Ingots are grouped into orders. Each order is made of one or more ingots with the same specifications. An order is characterized by:

- an order ID,
- a number of ingots,
- an alloy code,
- a product code,
- initial dimension (before hot rolling): thickness, width and length,
- final dimension (after hot rolling): gauge and width,
- a homogenization code,
- a due date.

As much as possible, ingots from the same order should be in the same batch and should be processed consecutively on the mill. We call each batch a *block*, and the orders in a block are sequenced according to their processing order on the hot mill. All ingots in a block must be rolled before another block can be processed on the same furnace and on the mill.

The due dates are handled indirectly by defining three different categories of orders: late, rush and normal. A late order is already late; a rush order has to be scheduled during the current roll life to avoid lateness; the remaining orders are normal.

2.4 Solutions

A solution to our problem corresponds to a sequence of blocks on the rolling mill that satisfies all hard constraints (and from which the sequences on the furnaces can be deduced). A solution also indicates the scheduling of each operation on the mill and furnaces.

3 Literature review

Literature on this type of problems is scarce. However, there are a little bit more publications for steel than for aluminum. Here are some important contributions that are related to our problem.

In the aluminum domain, Stauffer and Liebling [16] describe a problem similar to ours. Three furnace types are considered: pusher, large soaking pit and small soaking pit. To fill the soaking pits with minimum residual capacity, a bin-packing problem is solved. Alloys are split into groups of similar hardness, and each group has a wear coefficient and a feasible wear interval on the rolls. In this application, the rolling mill

does not run on a continuous basis, but is shut down every night and on Sunday. Also, width transitions are not taken care of. The objective considers both order tardiness and production quality (expressed through penalties). To solve this problem, the authors use a tabu search algorithm. A rough estimate of the minimum objective value is first calculated to quickly eliminate poor solutions and speed up the search. A rolling horizon approach is also developed to allow daily dynamic re-scheduling that takes into account new incoming orders and new priorities.

In [12], Lopez, Carter and Gendreau describe a tabu search approach to create hot strip mill production schedules in the steel industry. Long bars, called slabs, are first heated in one of two reheat furnaces. When slabs are at the right temperature, they can be rolled one by one on the roughing mill and then on the finishing mill. Reheat furnaces work in a continuous way, like pushers. Their goal is to create a roll life (sequence of slabs) to be processed on the finishing mill with minimum penalties associated with production quality, while insuring on-time deliveries and increased productivity. The penalties take into account factors similar to ours like width, gauge and hardness transitions. They also have different priority classes for the orders. The authors first develop a mathematical model for their problem, which can be seen a generalization of the PCTSP (Prize Collecting Traveling Salesman Problem). Then, a new idea called "canibalization" is introduced. Basically, the authors create several solutions with different slabs and combine good pieces taken from each sequence to generate better sequences. Different greedy heuristics are used to create starting solutions, which are then improved with a short tabu search. After the canibalization phase, longer tabu searches are performed to get the final solutions. The problem is similar to ours because pieces need to be selected and sequenced on the mill. On the other hand, they do not need to create blocks or batches for heating purposes because continuous furnaces are used.

Cowling and Rezig [5] propose a heuristic algorithm to solve a hot strip mill planning problem. The solution also integrates synchronization between the continuous caster that produce slabs and the mill. However, they do not mention homogenizing furnaces to heat slabs before hot rolling. Their mill is characterized by very strict width transitions and hardness constraints. A system of wear points is used to estimate roll wear. The objective is to maximize the "score" of a schedule, where the latter takes into account different factors like due date and production quality. The authors propose a mixed integer formulation of the problem. Their problem-solving approach iteratively fixes the values of the integer variables to obtain network flow subproblems that can be efficiently solved.

In [6], simple greedy heuristics are proposed to schedule a steel rolling mill. More sophisticated heuristic approaches, in particular genetic algorithms (GA), have also been applied to this kind of scheduling problems. Fang and Tsai [7] describe a GA to solve a steel hot strip mill scheduling problem with width transitions constraints ("coffin shapes") and roll wear. Tamaki et al. [17] develop an algorithm to schedule furnaces and a mill in a steel hot rolling process. To deal with their multi-criteria objective function, they exploit Pareto-optimal sets. Tang et al. [18] model a steel hot rolling mill as a multiple traveling salesman problem and solve it with a GA. In [14], Mayrand et al. propose another GA to schedule an aluminum cold mill with various technological constraints. Finally, Frontini et Bonhomme [8] describe a GA for an aluminum hot line in Brazil.

4 Algorithm

Our algorithm can be divided into three sequential steps. First, an insertion heuristic (H1) creates blocks while taking care to balance the workload on the furnaces. A first sequence of blocks on the mill is then created with another insertion heuristic (H2). Finally, this sequence is improved by eliminating some parts of it and by using heuristic H2 again to reconstruct it. The main steps of this algorithm are summarized in Figure 1.

Objective function

The objective is twofold and involves minimizing the hot mill idle time (in percent of total time) and the penalties associated with unsatisfied preferences. Due date considerations are included in the latter as

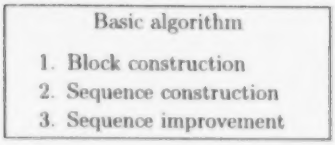

Figure 1: Main steps of the basic algorithm

"negative penalties" or rewards for scheduling orders on time (see below). The two objectives are weighted and aggregated into a single objective function f, using weighting parameters w_I for idle time and w_P for penalties, that is

$$f = w_P * Penalty + w_I * IdleTime. \quad (1)$$

The weighting parameters are very important because even small variations in their values can drive the search in a different region of the solution space. We exploit this property by generating many solutions using different weights (see Section 4.4, for more details).

Penalties

The standard penalties are those already mentioned in the previous section for modeling the preferences associated with the furnaces and the mill. Given that only a subset of orders can be scheduled during a roll life, orders that are left apart should also be considered when evaluating the quality of a schedule. Otherwise, some orders could be left apart forever because their inclusion would lead to a serious degradation of the current schedule. Consequently, there are also "negative penalties" or rewards that are used to force the method to include as much ingots as possible in the schedule and to meet the due dates.

On the hot mill, the transition penalties between two ingots are:

- penalty for one inch of width transition (wide to narrow: low penalty; narrow to wide: high penalty),
- penalty for one inch of thickness transition (thick to thin: low penalty; thin to thick: high penalty),
- penalty for alloy group transition (soft to hard: no penalty; hard to soft: high penalty).

On the furnaces, the penalties are:

- penalty per ingot if its homogenization code does not correspond to the furnace preferences,
- penalty per ingot if its homogenization code is not the same as the code of its batch.

The rewards are:

- negative value per ingot inserted in a furnace batch,
- negative value per ingot inserted in the hot mill sequence,
- negative value per late and rush order inserted in the hot mill sequence (the value is more negative for late order than for rush order).

The total penalty is thus the sum of penalties on the furnaces, penalties on the hot mill and rewards. Due to the latter, the total penalty can thus be negative.

4.1 Block construction

In the first phase, blocks of orders are created for the furnaces. These are the building blocks that will be used in the second phase to construct a sequence on the rolling mill. While constructing a block, penalties and rewards associated with the furnaces and the mill are considered. Consequently, blocks are not constructed blindly (i.e., with only furnaces in mind), as the schedule that follows on the mill is also considered. For a precise description in *pseudo-code* of the insertion heuristic H1, see Figure 2.

Procedure ConstructBlocks(K)

1. Divide orders into two classes: late and rush (C1) and normal (C2).
2. Divide each class into five subclass according to furnace preferences: pusher only, soaking pit only, preference pusher, preference soaking pit and no preference.
3. While the class C1 is not empty do
   a. Create a block: alternate the construction of blocks for the pusher and soaking pits to balance the workload (in hours).
   b. Create the list of available orders for the chosen furnace type.
   c. Select the order with smallest penalty to initialize the current block and remove this order from the list.
   d. Repeat
      i. If the list of orders is not empty do the following:
         A. For each order that can be feasibly inserted in the current block, determine the insertion place with minimum additional penalty.
         B. Among the orders with a penalty below threshold TH1, do the following:
            - Select the k^{th} best order (where k is a random value chosen between 1 and the minimum between parameter K and number of orders).
            - Insert the selected order at its best insertion place and remove it from the list.
      ii. If no order has been inserted in the block, exit repeat loop
4. For each block previously created
   a. Create the list of orders of class C2 that can be inserted in this block.
   b. Try Inserting orders from this list as in 3d.
5. Create new blocks with the remaining orders of class C2, as in 3.
6. Return the list of all created blocks.

Figure 2: Block construction (heuristic H1)

Procedure ConstructSequence(BlockList, w_P, w_I)

1. Create lists L1 and L2 from BlockList
2. Repeat
   a. If the list L1 is not empty do the following
      i. For each block that can be feasibly inserted in the current sequence, find the insertion place that leads to minimum objective value $f(CurrentSequence, w_P, w_I)$.
      ii. If at least one block qualifies then
         A. Select the best block according to this criterion.
         B. Insert the block at its best insertion place.
         C. Remove the selected block from L1.
         D. Return to step 2a.
   b. Try to insert one block from list L2 with additional penalty less than or equal to threshold TH2, as in step 2a.
   c. If no block from L2 is inserted, exit repeat loop.
3. Return the sequence

Figure 3: Sequence construction (heuristic H2)

First, the heuristic divides orders into two classes depending if they are "urgent" (i.e., late and rush) or "normal". Then, blocks are created with urgent orders only, while balancing the workload on the furnaces. To do this, the total heating and rolling time of all ingots in a batch are updated as orders are inserted. In our case, we want approximately twice as much hours on the soaking pits than on the pusher because there are two soaking pits and only one pusher. At each iteration, the next order to be inserted in the current block is randomly chosen among the K orders that introduce the lowest additional penalty. This randomization is aimed at introducing diversification into the block construction process (see the description of the multi-start strategy in Section 4.4). When inserting orders in a block, a threshold on the additional penalty is also defined to prevent the creation of bad blocks. When this threshold is reached for all orders or, when there are no more orders that can be feasibly inserted in the current block (due to furnace capacity constraints), the construction of a new block is started.

When all urgent orders have been processed, the blocks already created are filled with normal orders. This operation is aimed at fully exploiting the available capacity on the furnaces. Finally, new blocks are created with the remaining normal orders. It is worth noting that the blocks are associated with a furnace type at the end of this phase, but are not associated with a particular furnace (in the case of the soaking pits).

4.2 Sequence construction

In this second phase, a complete solution is created, that is, a sequence on the hot mill and a furnace assignment for each block is created. Starting from an empty sequence, blocks are selected from the pool created in Phase 1 and inserted one by one in the current sequence on the mill. To this end, two list of blocks are created: one with blocks containing at least one late or rush order (L1) and the other containing blocks with only normal orders (L2). For a precise description in *pseudo-code* of the insertion heuristic H2, see Figure 3.

This heuristic first tries to insert as many blocks from list L1 as possible. When this is not feasible anymore, blocks from list L2 that introduce an additional penalty less than or equal to threshold TH2 are considered for insertion. As soon as a block from L2 is inserted, blocks from list L1 are worked on again, because some of them can become feasible anew. This situation happens, for example, if an alloy group in L1

has a wear interval that does not start at 0. Assuming that the current sequence does not lead to sufficient wear on the roll to allow the insertion of orders from that alloy group, the addition of blocks taken from L2 will be considered to generate additional wear. This situation can also happen if the transition penalty between two blocks is too large and exceeds the threshold TH2. With the insertion of an "intermediary" block from L2, the width transition can become sufficiently "smooth" to be acceptable.

The assignment of furnaces to blocks is redone after the insertion of each block in the sequence, because the optimal assignment is likely to change. This reassignment is performed when the time schedule is updated and does not increase the overall complexity.

Creating a schedule from the sequence obtained is performed in the usual way. An availability list is maintained for every machine, and operations are performed as early as possible according to the sequence, while obeying the rule that a furnace is not available until all orders in the batch have been rolled.

4.3 Sequence improvement

To alleviate the impact of some potential bad choices made by the previous greedy construction heuristic, an improvement strategy has been implemented. The main steps are the followings:

1. Remove consecutive blocks in the sequence by following a specific "cutting" strategy,
2. Put back the removed blocks in the pool of unscheduled blocks,
3. Use heuristic H2 to insert blocks in the current sequence.

This procedure is repeated for a fixed number of iterations N_{impr}. At each iteration, a single random move is performed on the current sequence. If it leads to an improvement and the number of ingots in the new sequence contains at least 80% of the number of ingots in the previous sequence, the move is accepted and the resulting sequence becomes the current one. Otherwise, the move is rejected. This approach is a form of first improvement local hill climbing, although the neighborhood is not explored in a systematic way. As the evaluation of each move is costly, this approach is a good compromise between solution quality and computation time.

The cutting strategy can be implemented in many different ways. The following performed the best:

- select the cutting length at random between 1 and half the number of blocks in the sequence,
- select a cutting position at random between 1 and the number of blocks in the sequence minus the cutting length.

Figure 4 describes the sequence improvement procedure in pseudo-code.

4.4 Overall algorithm

The algorithm described in the previous sections is embedded within a two-level "multi-start" procedure. First, different sets of blocks are generated by exploiting the randomized order selection procedure in ConstructBlocks, using different values of K. Also, at the end, another set is obtained by collecting all blocks produced in the previous runs of ConstructBlocks. In the !atter case, duplicates are first eliminated; then, each time a block is inserted in the sequence, all blocks with one or more orders in common with the block just inserted are deleted. Second, with each set of blocks, a number of schedules are produced using different values around the desired ones for weighting parameters w_P and w_I in the objective.

The multi-start algorithm uses the following inputs:

- N_K: Number of sets of blocks generated with ConstructBlocks,
- K_i: Value of K used to generate set i=1,...,N_K,
- N_W: Number of values tried for weight pair (w_P,w_I),
- w_P: desired penalty weight value,
- w_I: desired idle time weight value,
- w_{Pj}: Penalty weight for try j=1,..,N_W (values near or equal to w_P),
- w_{Ij}: Idle time weight for try j=1,..,N_W (values near or equal to w_I),

Procedure ImproveSequence(Sequence, N_{impr}, w_P, w_I)

1. NumMaxIngots ← NumberOfIngots(Sequence)
2. BestSequence ← Sequence
3. Repeat N_{impr} times
   a. SequenceLength ← number of blocks in BestSequence
   b. CutLength ← RandomDiscretUniform(1, SequenceLength / 2)
   c. i ← RandomDiscretUniform(1, SequenceLength - CutLength + 1)
   d. j ← i + CutLength - 1
   e. CurrentSequence ← BestSequence with blocks i to j removed.
   f. Put back blocks i to j in the list of unscheduled blocks.
   g. Apply heuristic H2 on CurrentSequence.
   h. If $f(CurrentSequence, w_P, w_I) < f(BestSequence, w_P, w_I)$ and
      NumberOfIngots(CurrentSequence) >= 0.8 * NumMaxIngots then
         BestSequence ← CurrentSequence
         NumMaxIngots ← max(NumMaxIngots, NumberOfIngots(CurrentSequence))
4. Return BestSequence

Figure 4: Sequence improvement

- N_{impr}: Number of improvement iterations,

and calls the following procedures:

- ConstructBlocks(K): see figure 2,
- ConstructSequence(BlockList, w_P, w_I): see figure 3,
- ImproveSequence(Sequence, N_{impr}, w_P, w_I): see figure 4.

Figure 5 describes the complete algorithm in details. First, the algorithm creates N_K sets of blocks (plus the last one which is the union of all previously created block sets). Then, for each set, N_W sequences on the mill are obtained with different values of w_P and w_I, after applying both the construction and improvement procedures. At the end, the best sequence is returned.

5 Numerical results

In this section, numerical results are reported on test instances derived from real-world data. First, the general setting for these experiments is described in Section 5.1. Then, Section 5.2 studies the impact of threshold TH1, which is used to control the quality of blocks constructed in Phase 1. The impact of the multi-start and improvement strategies is analyzed in Section 5.3. Section 5.4 then reports computation times and solution quality obtained with an increasing number of orders. Finally, in Section 5.5, the algorithmic behavior over multiple roll lives is analyzed. All experiments were performed on an AMD Athlon 1.4Ghz processor with 1GB of RAM.

5.1 Settings

The general settings for our experiments are as follows.

Multi-start Algorithm

1. For i = 1 to $N_K + 1$
   a. If (i <= N_K) then
      BlockList$_i$ ← ConstructBlocks(K_i)
      Else
      BlockList$_i$ ← $\bigcup_{k=1}^{N_K}$ BlockList$_k$
   b. For j = 1 to N_w do
      i. CurrentSequence ← ConstructSequence(BlockList$_i$, w_{Pj}, w_{Ij})
      ii. CurrentSequence ← ImproveSequence(CurrentSequence, N_{impr}, w_{Pj}, $w_{I,j}$)
      iii. If $f(CurrentSequence, w_P, w_I) < f(BestSequence, w_P, w_I)$ then
         BestSequence ← CurrentSequence
2. Return BestSequence.

Figure 5: Multi-start algorithm

Furnace heating times

Two data sets were used for the furnace heating times, referred to as optimistic or pessimistic in the following. The optimistic times correspond to theoretical values provided by the furnace manufacturers. The pessimistic times come from averages, by homogenization codes and furnaces, obtained from historical data. The latter contain waste time (i.e., unnecessary waiting) due to occurrences of bad synchronization between a furnace and the hot mill. The "real" heating times thus fall somewhere between these two values. In the results that follow, optimistic times were used, unless otherwise specified.

Alloy groups

The maximum wear of a roll was set to 100. The wear intervals are:

- Group 1 (soft alloys): [0,50]
- Group 2 (hard alloys): [0, 100]

Penalties

The values for the different types of penalties are given in Table ??.

Thresholds

Threshold TH2 has always been set to 0. Hence, blocks made of normal orders only are considered for insertion in the sequence when the additional penalty is less than or equal to 0 (i.e., they must provide some benefits). For blocks with rush or late orders, this threshold does not apply. In the case of threshold TH1, experiments have been performed to identify good values, as described below.

Weighting parameters

In all experiments, we have $w_P + w_I = 1$, with either w_P=0.003 or 0.00001. In the first case, equal importance is given to penalties and idle time (w_P is small because the two objectives are not on the same scale). In the second case, idle time is favored.

penalty for one inch of width transition (wide to narrow)	10
penalty for one inch of width transition (narrow to wide)	50
penalty for one inch of thickness transition (thick to thin)	10
penalty for one inch of thickness transition (thin to thick)	50
penalty for alloy group transition (soft to hard)	0
penalty for alloy group transition (hard to soft)	500
penalty per ingot if its homogenization code does not correspond to the furnace preferences	200
penalty per ingot if its homogenization code is different from the code of its batch	50
negative value per ingot inserted in a furnace batch	-10
negative value per ingot inserted in the hot mill sequence	-10
negative value per rush order inserted in the hot mill sequence	-100
negative value per late order inserted in the hot mill sequence	-1000

Table 1: Penalties

5.2 Impact of Threshold TH1

The goal here is to find good values for threshold TH1. To perform this calibration, the settings in Table 2 were used.

# Exp.	w_P	w_I	N_W	w_{P1}	w_{I1}	N_K	K_1	N_{impr}
1	0.003	0.997	1	0.003	0.997	1	1	0
2	0.00001	0.99999	1	0.00001	0.99999	1	1	0

Table 2: Settings for evaluating the impact of threshold TH1

Tables 3 and 4 show the results for the two sets of parameters on different instances of type $x/y/z$, where x is the number of normal orders, y the number of rush orders and z the number of late orders that are available for constructing a schedule. Figures 6 and 7 are a graphical representation of the same results. We note that good values for TH1 depend on the weighting parameters. Based on these results, we thus set TH1 to 100 when $w_P = 0.003$ and to 500 when $w_P = 0.00001$. This makes sense, as we are certainly ready to accept much larger penalties in the second case, when the emphasis is on idle time.

5.3 Impact of multi-start and improvement strategies

Here, we study the sensitivity of our algorithm to parameters N_K, N_W and N_{impr}. As these parameters strongly interact, we designed a "three-dimensional" experiment which considers all of them at once. We used the 150/50/10 problem instances for these tests.

Table 5 shows the results obtained with different parameter values. Each entry in this table is the average of the objective value over 10 runs. It should be noted that the comparison is made on the basis of solution quality only, as the longest runs (those with the largest parameter values) take less than 10 minutes of CPU time, which is largely acceptable in our context.

To evaluate the impact of our multistart strategy, we can look at the results obtained with w_P=0.00001. First, the objective clearly improves for $N_W = 1$ when N_K is increased from 1 to 10+all (where "all" means that the union of all sets of blocks generated in the previous iterations is also considered, as explained in Section 4.4). In some cases, further improvement is obtained by setting N_K to 20+all, but this is relatively marginal. Second, by increasing N_W from 1 to 10, the objective value also improves, with some additional improvement obtained with $N_W = 20$. To take a practical example, with $N_K = 1$ and $N_{impr} = 100$, the objective can be reduced from 46.9 to 40.5 by increasing N_K to 10+all. Then, by increasing N_W from 1 to 10

TH1	Prob 1 50/25/0	Prob 2 100/25/10	Prob 3 150/25/10	Prob 4 50/50/0	Prob 5 100/50/10	Prob 6 150/50/10
-500	70.2	39.6	39.0	74.2	41.4	41.4
-200	70.2	39.6	39.0	74.1	39.7	39.7
-100	59.5	18.8	18.8	55.0	24.4	24.4
-50	57.6	20.7	14.9	51.9	21.5	21.5
0	53.1	15.5	14.9	52.3	20.1	15.6
50	45.8	18.4	17.2	52.6	17.5	13.7
100	45.1	18.5	16.4	53.2	17.2	13.1
200	51.1	17.6	18.2	52.5	18.8	16.0
300	44.6	20.7	22.2	51.8	21.2	22.8
400	44.6	27.0	25.8	46.8	22.7	21.9
500	46.1	27.5	23.6	50.7	19.9	17.9
600	46.1	22.4	23.6	55.5	19.9	17.9
700	46.1	22.4	23.6	58.3	25.7	21.5
800	46.1	22.4	23.6	60.0	22.4	21.5
900	68.2	26.9	26.2	62.1	22.4	21.5
1000	68.2	26.9	26.2	62.1	22.4	24.5
2000	68.2	26.9	26.2	62.1	22.4	24.5
5000	72.9	31.6	26.2	65.7	22.4	24.5
10000	72.9	31.6	26.2	65.7	22.4	24.5

Table 3: Impact of threshold: experiment 1 ($w_P = 0.003$)

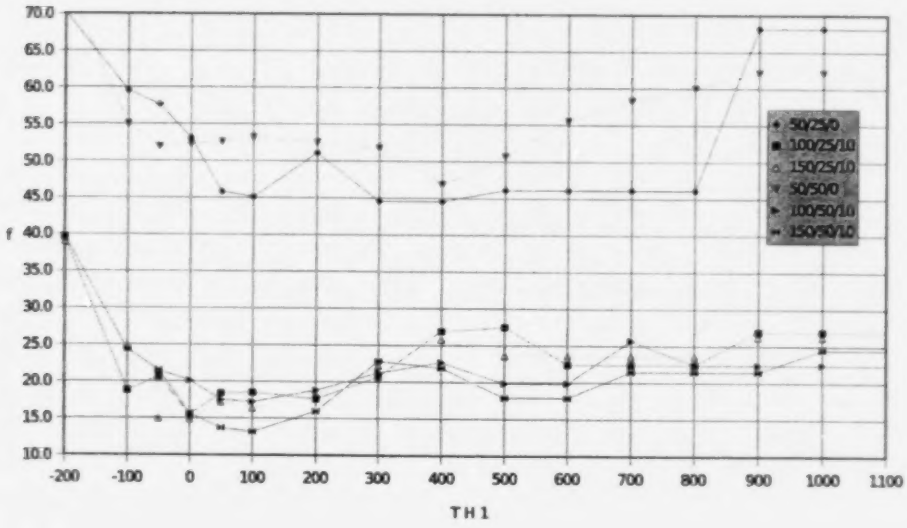

Figure 6: Impact of threshold: experiment 1 ($w_P = 0.003$)

TH1	Prob 1 50/25/0	Prob 2 100/25/10	Prob 3 150/25/10	Prob 4 50/50/0	Prob 5 100/50/10	Prob 6 150/50/10
-500	86.3	78.2	78.2	85.1	84.3	84.3
-200	86.3	78.2	78.2	85.2	83.6	83.6
-100	66.6	53.6	52.7	71.2	64.4	64.4
-50	61.8	57.2	52.1	65.0	63.1	63.1
0	62.1	54.9	52.5	64.5	59.8	57.4
50	55.6	54.5	53.2	64.7	57.9	55.4
100	72.7	54.7	53.2	64.7	57.9	53.5
200	53.6	54.7	53.2	64.5	57.9	53.5
300	51.1	51.8	53.2	64.5	60.0	56.0
400	51.1	51.9	53.2	61.6	56.0	47.3
500	51.2	47.0	51.7	61.6	56.6	47.3
600	51.2	50.3	50.3	59.4	56.6	47.3
700	51.2	47.2	50.4	61.5	53.4	47.5
800	51.2	47.2	50.4	59.4	55.6	47.7
900	67.1	50.6	52.1	59.5	55.6	43.8
1000	67.1	50.6	52.1	59.5	55.6	43.8
2000	67.1	50.6	52.1	59.5	55.6	43.8
5000	65.4	51.4	52.1	55.6	55.6	43.8
10000	65.4	51.4	52.1	55.6	55.6	43.8

Table 4: Impact of threshold: experiment 2 ($w_P = 0.00001$)

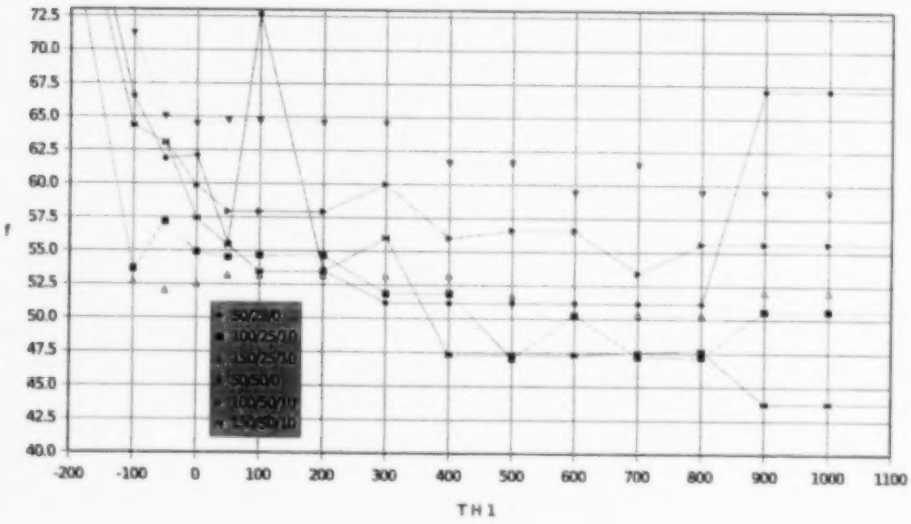

Figure 7: Impact of threshold: experiment 2 ($w_P = 0.00001$)

TH1	Prob 1 50 25 0	Prob 2 100 25 10	Prob 3 150 25 10	Prob 4 50 50 0	Prob 5 100 50 10	Prob 6 150 50 10
500	70.2	59.6	59.0	74.2	41.4	41.4
-200	70.2	59.6	59.0	74.1	49.7	49.7
100	58.5	18.8	18.8	55.0	24.4	24.4
50	57.0	20.7	14.9	51.9	21.5	21.5
0	53.1	15.5	14.9	52.4	20.1	15.6
50	45.8	18.4	17.2	52.6	17.5	14.7
100	45.1	18.5	16.4	54.2	17.2	13.4
200	51.1	17.6	18.2	52.5	18.8	16.6
300	44.6	20.7	22.2	51.8	21.2	22.8
400	44.6	27.9	25.8	46.5	22.7	21.9
500	46.1	27.5	23.6	50.7	19.9	17.9
600	46.1	22.4	23.6	52.5	19.9	17.9
700	46.1	22.4	23.6	58.4	25.7	21.5
800	46.1	22.4	23.6	60.0	22.4	21.5
900	68.2	26.9	26.2	62.1	22.4	21.5
1000	68.2	26.9	26.2	62.1	22.4	21.5
2000	68.2	26.9	26.2	62.1	22.4	21.5
5000	72.9	31.6	26.2	65.7	22.4	21.5
10000	72.9	31.6	26.2	65.7	22.4	21.5

Table 3: Impact of threshold: experiment 1 (w_f = 0.001)

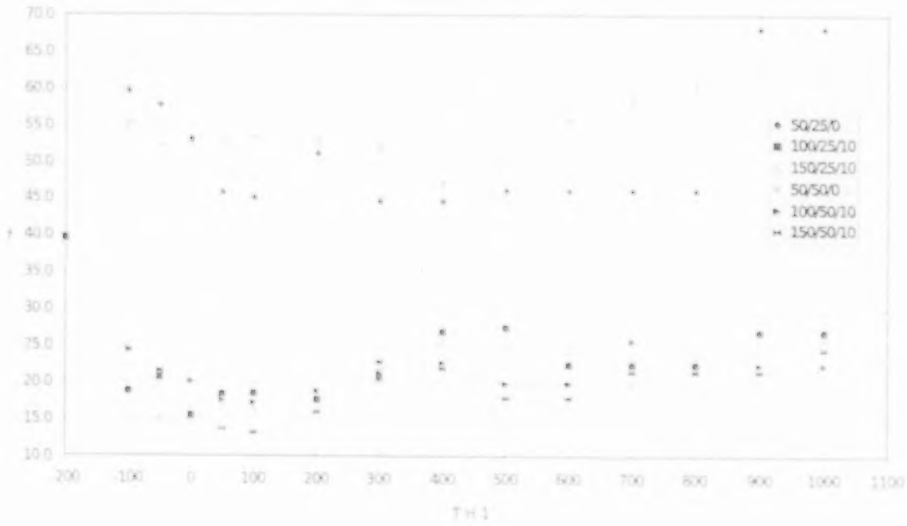

Figure 6: Impact of threshold: experiment 1 (w_f = 0.001)

TH1	Prob 1 50 25 0	Prob 2 100 25 10	Prob 3 150 25 10	Prob 4 50 50 0	Prob 5 100 50 10	Prob 6 150 50 10
-500	86.3	78.2	78.2	85.1	84.6	84.6
-200	86.3	78.2	78.2	85.2	85.6	84.6
-100	86.6	84.6	82.7	81.2	80.4	84.4
-50	61.8	57.2	82.1	65.0	60.1	63.1
0	62.1	54.9	82.5	64.5	59.8	62.4
50	55.6	54.3	53.2	61.7	57.9	55.4
100	72.7	54.7	53.2	64.7	57.9	55.4
200	53.6	54.7	53.2	64.5	59.9	59.5
300	51.1	51.8	53.2	64.5	60.0	56.6
400	51.1	51.9	53.2	61.6	56.0	47.3
500	51.2	47.0	51.7	61.6	50.0	47.3
600	51.2	50.4	50.4	58.4	56.6	47.3
700	51.2	47.2	50.4	61.7	53.4	47.7
800	51.2	47.2	50.4	58.4	55.6	47.7
900	67.1	50.6	52.1	58.7	52.6	44.8
1000	67.1	50.6	52.1	58.7	52.6	44.8
2000	67.1	50.6	52.1	58.7	52.6	44.8
5000	65.4	51.4	52.1	55.6	52.6	44.8
10000	65.4	51.4	52.1	55.6	52.6	44.8

Table 4: Impact of threshold: experiment 2 (avg. turnout)

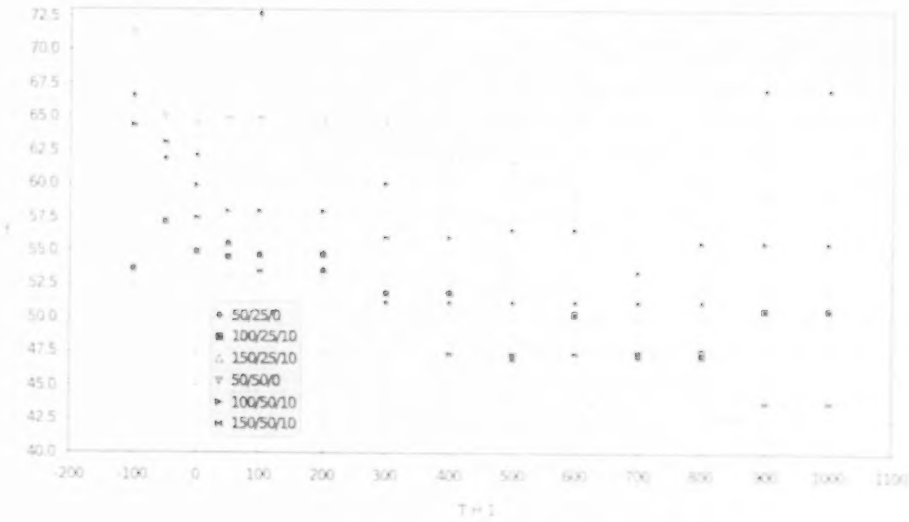

Figure 7: Impact of threshold: experiment 2 (avg. turnout)

$N_W = 1$	w_P=0.00001					w_P=0.003				
$N_{impr} \setminus N_K$	1	5	5+all	10+all	20+all	1	5	5+all	10+all	20+all
0	47.3	45.0	44.2	43.5	43.5	21.9	18.9	18.1	16.8	16.0
5	47.3	43.8	43.6	43.3	43.1	19.6	16.8	16.6	13.9	13.8
10	47.3	44.3	43.6	43.3	42.6	19.3	16.1	14.2	12.4	13.1
25	47.3	43.0	42.6	41.0	41.8	19.2	15.0	12.7	11.5	12.3
50	47.0	43.1	41.3	40.9	40.2	19.0	13.3	13.3	12.4	10.0
100	46.9	42.5	40.5	40.5	39.1	19.0	14.2	13.1	11.5	11.4
200	47.0	41.8	40.1	38.7	38.9	19.0	14.0	12.4	12.6	11.5
$N_W = 5$	w_P=0.00001					w_P=0.003				
$N_{impr} \setminus N_K$	1	5	5+all	10+all	20+all	1	5	5+all	10+all	20+all
0	47.3	44.6	45.3	43.8	43.2	19.2	17.5	15.9	14.8	13.9
5	47.3	44.1	43.4	42.4	42.7	13.8	13.9	13.4	11.8	11.4
10	47.3	43.2	42.8	42.8	40.6	13.9	12.3	11.1	10.8	10.5
25	47.3	43.0	41.1	40.4	40.4	10.5	9.8	9.5	9.5	8.5
50	45.7	41.1	39.9	38.9	37.6	11.8	9.4	8.7	8.5	8.4
100	46.0	42.3	39.0	38.7	37.3	9.8	8.8	8.6	8.0	8.0
200	45.5	39.8	37.4	36.7	36.4	9.7	9.0	7.9	7.1	7.0
$N_W = 10$	w_P=0.00001					w_P=0.003				
$N_{impr} \setminus N_K$	1	5	5+all	10+all	20+all	1	5	5+all	10+all	20+all
0	47.3	45.9	44.2	43.5	43.5	19.2	16.3	17.4	14.5	13.9
5	47.3	43.6	43.1	42.2	42.4	12.0	10.4	10.2	10.6	10.0
10	47.3	43.4	43.2	41.7	39.5	9.8	9.8	9.4	9.7	8.9
25	46.0	40.1	40.4	38.4	38.2	8.6	9.6	8.5	8.4	8.3
50	45.6	40.1	37.6	37.5	36.6	8.3	8.1	7.7	8.0	7.4
100	45.1	38.8	38.2	36.6	37.0	8.1	8.2	7.6	7.3	7.1
200	43.3	40.6	37.2	36.3	36.5	8.1	7.9	7.2	7.7	6.7
$N_W = 20$	w_P=0.00001					w_P=0.003				
$N_{impr} \setminus N_K$	1	5	5+all	10+all	20+all	1	5	5+all	10+all	20+all
0	47.3	44.6	44.9	43.4	43.2	19.2	16.9	16.1	15.3	14.6
5	47.3	43.5	41.7	42.7	40.5	9.7	9.3	10.0	9.2	9.5
10	47.0	42.7	42.0	41.3	39.5	9.5	8.8	8.6	9.5	8.5
25	47.0	39.8	40.2	37.3	37.4	8.2	8.2	8.0	7.7	7.9
50	44.2	40.0	39.0	36.8	36.8	8.1	8.2	7.7	7.6	6.9
100	43.3	38.7	37.4	36.5	36.3	8.1	8.0	7.5	7.0	7.5
200	42.2	39.3	36.7	36.5	36.3	7.8	8.1	7.4	6.9	6.7

Table 5: Impact of multi-start and improvement

	w_P=0.00001		w_P=0.003	
Orders	f	Time (s)	f	Time (s)
75/25/5	44.7	71	26.1	81
150/50/10	36.7	51	7.6	52
300/100/20	28.4	121	-31.9	130
450/150/30	26.8	181	-51.7	188
600/200/40	24.9	228	-67.9	241

Table 6: Impact of increasing number of orders

the objective is further decreased to 36.6. Clearly, parameter N_{impr} is also important. With $N_{impr} = 0$, N_K = 10+all and $N_W = 10$ a value of 43.5 is reached, instead of 36.6. Thus, the multi-start and improvement strategies act in synergy to provide additional benefits.

Based on the results obtained, N_K = 10+all, $N_W = 10$ and $N_{impr} = 100$ (for both values of w_P) were chosen in the following experiments.

5.4 Increasing number of orders

Here, we want to examine the impact of problem size on solution quality and computation time. Table 6 shows results obtained with an increasing number of normal, rush and late orders. Each objective value f is an average taken over 10 different runs.

One odd result is related to the smaller computation times observed when the number of orders is increased from 75/25/10 to 150/50/10. In fact, we discovered that the computation times of the 75/25/10 instance are larger than they should be. As a typical sequence on the mill contains approximately 75 orders, most orders are found in the final sequence on the mill. Due to threshold TH1 and a limited choice of orders, blocks are not filled much and there are a relatively large number of them. The block insertion operation is thus more computationally expensive. We have observed the same phenomenon on three other problem instances of the same size. When the number of orders is substantially increased, this effect disappears as many different alternative orders can be chosen during block construction. Consequently, most blocks are filled or almost filled and those are the ones that are used to generate the sequence on the mill.

The same observation explains the larger computation times obtained with w_P=0.003, when compared with w_P=0.00001. As threshold TH1 is lower in the former case, there are more partially filled blocks and the sequence on the mill is made of a larger number of those blocks. This effect is particularly obvious on the smaller problem instances 75/25/10 and 150/50/10.

Finally, we observe that solution quality increases with problem size. This is not surprising, given that a larger number of orders are available to create and fill blocks (i.e., there are more choices).

5.5 Multiple roll lives

The purpose of this section is to evaluate if our algorithm is effective on the long run, over multiple roll lives. We want to be sure that no degradation in quality is observed from the first rolls to the last ones. For example, this could happen if our algorithm would only focus on "good" orders to schedule the first rolls, while leaving apart difficult ones. The latter would tend to accumulate over time, thus leading to poor schedules at the end. We used both optimistic and pessimistic heating times for these experiments (see section 5.1).

Initially, we start the simulation with a scheduling pool made of a 150/50/10 mix of orders. After each roll life, new orders are added to the pool, depending on the number of days elapsed since the beginning of the roll cycle. The distribution of these new orders with each elapsed day is chosen to keep the plant in equilibrium (i.e., the number of new orders added to the pool is approximately equal to the number of orders found in a sequence on the mill). This way, the pool size never gets too large nor too low.

w_P=0.00001	Optimistic heating time				Pessimistic heating time			
Roll no	f	Penalty	Idle Time	Orders	f	Penalty	Idle Time	Orders
1	36	-7233	36.4	70	42	-5032	42.7	69
5	35	-320	35.9	69	41	-1093	41.2	70
10	40	-5001	40.5	66	60	415	60.1	81
15	41	-8472	41.1	97	43	-2329	43.1	70
20	45	-6487	46.0	76	51	-490	51.2	77
25	56	-12306	57.0	74	49	-2363	49.2	80
30	38	-1530	38.5	66	58	-9198	58.3	74
35	31	1146	31.1	54	59	-10827	59.3	76
40	39	-1479	39.5	81	55	-161	55.8	82
45	35	-2607	35.4	64	53	-1342	53.4	70
46	53	-8.84	53.5	89	49	-12051	49.8	81
47	34	-10545	34.7	66	54	-19084	54.9	82
48	43	1628	43.4	79	57	-2156	57.4	79
49	46	-652	46.7	81	43	-10091	43.3	80
50	31	-13434	32.1	87	64	-27039	64.9	85
Average	41.5	-1816	41.5	74.6	53.0	-4681	53.0	75.8
Minimum	26	-13434	26.1	53	41	-27039	41.2	58
Maximum	56	8235	57.0	97	64	3995	64.9	97
Std Dev	7.3	4521	7.3	10.7	6.0	5596	6.0	8.2

Table 7: Multiple roll lives (w_P=0.00001)

After one roll life, the time of the last use of each machine is recorded and becomes the new availability time for the next roll life. The time needed to change a roll is also taken into account when the hot mill capacity is evaluated. Tables 7 and 8 present results obtained with w_P=0.00001 and w_P=0.003, respectively, over 50 roll lives. Only the values associated with a few roll schedules are shown along the way, but the last 6 roll lives are reported in the Tables. At the end of the Tables, the average, minimum, maximum and standard deviation of the objective over the 50 roll lives are reported.

Clearly, solution quality does not degrade over time. For example, with optimistic heating times and w_P=0.00001, 3 values out of the last 6 are below the average objective value. Optimistic or pessimistic heating times substantially impact the idle time percentage. Indeed, the average increases from about 41.5 to 53.0 for w_P=0.00001 and from 45.8 to 56.3 for w_P=0.003, depending if optimistic or pessimistic heating times are used, respectively. In all cases, including those with pessimistic heating times,the idle time is substantially lower than what is observed in the "real" schedules (where 50 roll lives correspond to approximately one year of production). The comparison with real schedules must be done cautiously, however, because (1) orders for the simulation were generated according to general trends observed in historical data, as the true orders were not all available to us and (2) penalties for undesirable transitions in the sequence might be a gross approximation of what is enforced and implemented in practice on the floor. It remains that the schedules produced by our heuristic look very competitive with those implemented in practice.

6 Conclusion

In this paper, we have presented a heuristic algorithm to schedule the furnaces and the mill in a hot rolling plant. Standard technological constraints are taken into account like hardness transitions, roll wear, homogenization code compatibility and width/thickness transitions. The heuristic minimizes the idle time on the mill plus soft constraint penalties related to production quality. Numerical results on problem instances derived from real world data are reported. These results show the benefits of the multi-start and improvement strategies embedded within the heuristic. Experiments over 50 consecutive roll lives have also shown that

w_P=0.003	Optimistic heating time				Pessimistic heating time			
Roll no	f	Penalty	Idle Time	Orders	f	Penalty	Idle Time	Orders
1	6	-12025	43.1	60	13	-11498	48.4	61
5	32	-1707	37.9	61	47	-5859	65.4	56
10	44	-2887	53.7	78	52	-2950	61.2	64
15	42	-3703	53.3	88	40	-7527	63.3	68
20	44	-1034	48.0	83	51	-4476	64.6	60
25	21	-6534	41.5	80	26	-12083	63.1	78
30	30	-3126	40.2	70	33	-7657	56.1	93
35	16	-6666	36.1	70	37	-4002	49.7	66
40	36	-4443	49.8	82	8	-10932	41.8	82
45	39	-6124	58.5	89	23	-10552	55.6	81
46	22	-5340	38.7	69	23	-11910	59.4	64
47	43	-1318	47.5	68	39	-6574	59.0	70
48	32	-4641	46.2	64	29	-8423	55.1	62
49	20	-12162	57.3	76	17	-17603	70.2	83
50	24	-511	25.9	76	14	-7850	37.6	84
Average	29.8	-5316	45.8	74.6	32.0	-8025	56.3	74.7
Minimum	-15	-15584	22	57	-29	-29404	38	56
Maximum	52	1296	61	90	60	550	74	101
Std Dev	12.3	3897	8.6	8.6	18.3	5793	7.9	11.3

Table 8: Multiple roll lives (w_P=0.003)

no degradation is observed in the schedule over time. In addition, the idle time is substantially lower than what is observed in real schedules, even when pessimistic heating times are used.

One interesting extension to this work would be to model the pusher as a furnace that can be continuously fed with new ingots over time (rather than as a "batch" furnace). Other developments would be related to the integration of more complicated production processes associated with "brazing products". In this case, the central part (core) must first be produced, followed by the top and bottom plates. Then, the three parts must be assembled together. Clearly, this type of process makes the optimization problem much more difficult.

Acknowledgments

This research was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) and by Keops Technologies Inc. Their support is gratefully acknowledged.

References

[1] Baptiste, P. *Batching Identical Jobs.* Mathematical Methods of Operation Research, Toulouse, 52:355-367, 2000.

[2] Brucker, P., Gladky, A. et al. *Scheduling a batching machine*, Journal of Scheduling 1:31-54, 1998.

[3] Ralf B. *Scheduling.* Handbook of Evolutionary Computation release 97/1, IOP Publishing Ltd and Oxford University Press, 1997.

[4] Chen, T.C.E. and Sin, C.S.S. *A state-of-the-art review of parallel-machine scheduling research.* European Journal of Operational Research. 47:271-292, 1990.

[5] Cowling, P. and Rezig, W. *Integration of continuous caster and hot strip mill planning for steel production.* Journal of Scheduling, 3:185-208, 2000.

[6] Dupont, L., Rézard, H. and Allaili, A. *Ordonnancement sous contraintes d'un laminoir à chaud.* Deuxième Congrès International Franco-Québécois de Génie Industriel - ALBI, 1997.

[7] Fang, H.-L. and Tsai, C.-H. *A Genetic Algorithm Approach to Hot Strip Mill Rolling Scheduling Problems.* IEEE (0-7803-5214-9/98), 1998.

[8] Frontini. G. F. and Bonhomme, J. *Optimizing Asset Utilization in Aluminum Hot Rolling Mills.* 3ième Congrès international de génie industriel, École Polytechnique de Montréal, 1999.

[9] Glass, C. A. and Potts, C. N. *Chapter 11 : Machine scheduling.* Local Search in Combinatorial Optimization, p.361-414, John Wiley & Sons Ltd, 1997.

[10] Hoon Lee, Y. and Pinedo, M. *Scheduling jobs on parallel machines with sequence-dependent setup times.* European Journal of Operational Research, 113:464-474, 1997.

[11] Jain, A.S. and Meeran, S. *Deterministic job-shop scheduling : Past, present and future.* European Journal of Operational Research, 113:390-434, 1999.

[12] Lopez, L., Carter M. W. and Gendreau, M. *The hot strip mill production scheduling problem : A tabu search approach.* European Journal of Operation Research, 106:317-335, 1998.

[13] Lopez, P., Roubellat, F. et al. *Ordonnancement de la production*, Hermes, 2001.

[14] Mayrand, É., Lefrançois, P., Kettani, O. and Jobin, M.-H. *A genetic search algorithm to optimize job sequencing under a technological constraint in a rolling-mill facility.* OR Spektrum 17:183-191, 1995.

[15] Reklaitis, G. V. *Overview of planning and scheduling technologies.* Latin American Applied Research, 30:285-293, 2000.

[16] Stauffer, L. and Liebling, Th. M. *Rolling horizon scheduling in a rolling-mill.* Annals of Operations Research 69:323-349, 1997.

[17] Tamaki, H., Mori, M. and Araki, M. *Multi-criteria optimization by genetic algorithms : A case of scheduling in hot rolling process.* Proceedings of APORS'94, pp. 374-381, World Scientific Publishing, 1994.

[18] Tang, L., Liu, J., Rong, A. and Yang, Z. *A multiple traveling salesman problem model for hot rolling scheduling in Shanghai Baoshan Iron & Steel Complex.* European Journal of Operation Research 124:267-282, 2000.

[19] Vaessens, R.J.M., Aarts, E.H.L. and Lenstra, J.K. *Job Shop Scheduling by Local Search.* INFORMS Journal on Computing 8:302-317, 1996.

[20] Vignier, A., Brillaut, J.-C. and Proust, C. *Les problèmes d'ordonnancement de type flow-shop hybride: état de l'art.* RAIRO Recherche Opérationnelle 33:117-183, 1999.

[21] Yang, W.-H. and Liao, C.-J. *Survey of scheduling research involving setup times.* International Journal of Systems Science 30:143-155, 1999.